



## Projet 6 POO- Routing - PHP - Réservation de gîtes

Projet seul ou en groupes - Temps estimé : **15-20 jours**

Technologies, outils et concepts travaillés : PHP, programmation orientée objet (POO), HTML, CSS, JavaScript, (Sass, jQuery), UML, GIT, GITHUB

Ressources :

<https://github.com/michelonlineformapro/Projet-6-Poo-Gites-Update-Router>

### Description du projet :

Vous allez réaliser une plate forme de réservation de gîtes se situant dans une zone géographique au choix).

- La plate forme est gérée par un seul administrateur qui pourra s'occuper d'ajouter, modifier ou supprimer des gîtes et gérer des utilisateurs, Le backend CRUD Administrateur est sécurisé par une session pour interdire les accès a certaines pages.
- Côté utilisateur : on demande une inscription et une connexion pour réserver un gîte et ajouter des commentaires de satisfaction.
- Côté visiteur : on pourra consulter la liste des différents gîtes ainsi que leurs détails.
- On pourra aussi effectuer une recherche pour trouver un gîte selon ses critères et ses disponibilités. (Date d'arrivée, date de départ, nombre de chambre, nombre salle de bain, zone géographique, Type de logement, etc...).

Il vous faudra créer une classe (Modèle) représentant un hébergement.

Il existera plusieurs catégories d'hébergements (exemple : chambre, appartement, maison, villa...) = clé étrangère.

Chaque catégorie d'hébergement peut avoir des spécificités, par exemple pour un appartement ou un camping, on peut indiquer le nombre de pièces, pour une maison la présence d'un jardin particulier, douche, etc.

#### Côté administrateur :

##### Un back-office

- De visualiser l'ensemble des gîtes et leur statut (occupé ou libre)
- D'ajouter des gîtes
- De supprimer des gîtes
- De les éditer

#### Côté utilisateur :

- Réserver un gîte
- Poster des commentaires

#### Côté visiteur :

- Une page accueil avec :
  - La liste des gîtes disponibles
  - Un formulaire de recherche (date de départ, date de fin, nombre de couchages, Région, etc...)
- Une fiche par gîte avec un formulaire de réservation et la possibilité d'ajouter des commentaires (connexion obligatoire)

#### Gestion des disponibilités :

- Niveau 1 :
  - Lorsqu'un gîte est réservé il devient indisponible et ne plus apparaître dans les recherches. Une fois disponible, on peut le réserver à nouveau
- Niveau 2 :
  - Lorsqu'un gîte est réservé pour une période il devient indisponible pour celle-ci. Il reste accessible par une recherche mais il faudra indiquer les jours d'indisponibilité. Il est possible de faire une réservation pour les jours disponibles.

#### *Les plus : à réaliser si vous avez le temps*

- Afficher un calendrier de disponibilité de chaque gîte
- Afficher la localisation des gîtes sur une carte (openstreetmap + leaflet)
- Créer un espace membres pour les clients
- Valider une inscription au site par email
- Animation Javascript / jQuery
- Sass + framework CSS (tailwind, materialize, bulma, etc...)

### Rappel des tâches :

**Le benchmarking :** <https://fr.wikipedia.org/wiki/Benchmarking>

1. Benchmark + wireframe et/ou maquette du site (figma, adobeXD, etc ...)
2. Écrire un product Backlogs via github → projects
3. Concevoir un MCD (Modèle Conceptuel de Données)
4. Concevoir un MLD (Modèle Logique de Données)
5. Concevoir le MPD (Modèle Physique de Données)
6. Concevoir un diagramme de séquence
7. En cas de groupe : Se répartir les taches pour la réalisation du back-end & front-end
8. Créer un dépôt GitHub, effectué des commits régulier et fusionner (merge) les commit en cas de travail en groupe

### Objectifs :

- Découvrir la programmation orientée objet en PHP
- Consolider les bases du CRUD
- Consolider/approfondir les bases du langage SQL avec des appels de clé étrangère et des requêtes SQL robuste

### Thèmes :

- Programmation orientée objet
- Bases de données
- Envoi et confirmation par email

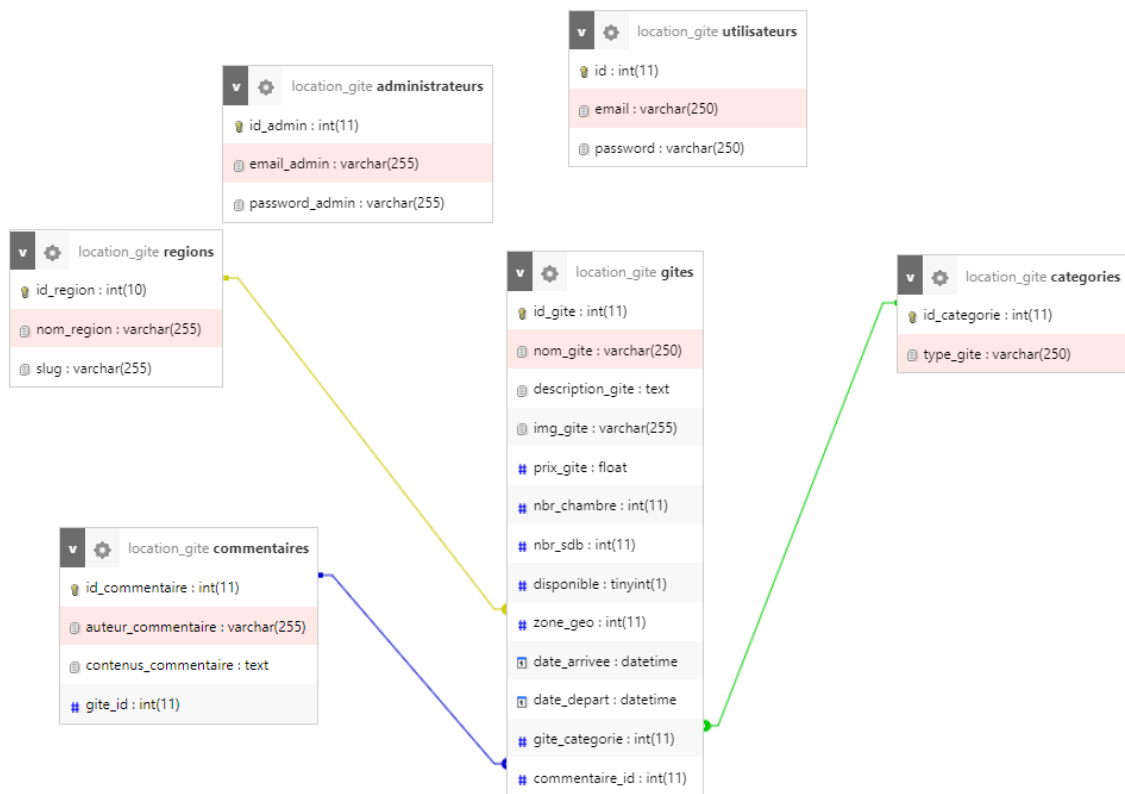
### Ressources :

Parcours "PHP > Programmation orientée objet" sur [vos-formations.com](https://vos-formations.com)

- <https://www.pierre-giraud.com/php-mysql-apprendre-coder-cours/introduction-programmation-orientee-objet/>
- <https://www.grafikart.fr/formations/programmation-objet-php>
- <https://sql.sh/>

# Étapes :

Votre MCD (vous pouvez faire le vôtre différemment)



## BACKEND :

- 1- Créer un Template (Menu (navbar) + conteneur principale + footer)
- 2- Créer un système de routing pour des URLs propres
- 3- Créer des classes et des méthodes pour séparer la couche métier du contenu visuel
- 4- Votre espace administration
  - a. Dans la navbar -> connexion -> formulaire de connexion sécurisé
  - b. Une fois connecter le bouton connexion disparaît et devient déconnexion et laisse apparaître l'onglet administration
  - c. Utilisation de la variable super Globale \$\_SESSION ['connecter'], des condition if {} else {} pour vérifier la connexion et interdire de page
- 5- Une fois connecter créer votre CRUD Gîtes avec des redirections ou des fenêtres modales

- a- Soit un formulaire de connexion pour l'administrateur et les utilisateurs sur 2 tables différentes
- b- Connexion par email (et / ou pseudo) et mot de passe
- c- Créer une classe Administration et une méthode connexionAdmin(), inscription() et connexionUser()
- d- Créer des instances de la classe administration dans la page connexion et faire appel aux méthodes loginAdmin () et loginUsers()
- e- Faire une redirection vers votre page d'accueil du CRUD
- f- Réaliser les 5 opérations CRUD des gîtes, à l'aide d'une classe Gîtes
- g- Un système d'inscription pour les utilisateurs (éventuellement une validation par email)
- h- Un système de réservation de gîte + confirmation par email

## FRONTEND:

- 1- Un moteur de recherche de gîte par nom, date arrivée, date de départ et nombre de chambre, nombre de salle de bain et par fourchette de prix
- 2- Une page d'accueil avec tous les gîtes disponibles (grisé les gîtes non dispos ou les cachés)
- 3- Réaliser une requête SQL avec une variable date départ < date du jour
- 4- Afficher seulement le bouton détails
- 5- Sur la page de détails ajouter un bouton réserver
- 6- Créer un formulaire de réservation (Email au minimum + PAS OBLIGATOIRE = nom, prénom, adresse etc...)
- 7- Créer une classe PHP Mailer qui envoie un email à mailTrap
- 8- Renvoyer votre formulaire de réservation vers l'instance de la classe PHP Mailer et sa méthode d'envoi
- 9- Dans le gabarit email, ajouter toutes les infos du gîte et un bouton confirmer la réservation
- 10- Quand le gîte est réservé il devient grisé ou disparaît jusqu'à l'échéance de la date de départ
- 11- Réaliser une redirection vers la page d'accueil

## RAPPELS :

### La programmation orientée objet :

- Le code que vous développez répondra à certaines contraintes et certains besoins.
- Structurez l'ensemble de votre code pour le rendre plus solide et facile à entretenir ou à faire évoluer.

- Être capable de faire les bons choix de structuration, logique et organisation de votre code vous différencie en tant que développeur

### Les objets et les classes :

- En programmation orienté objet, une **classe** est un modèle ou un mode d'emploi tandis que les **objets** sont les objets créés à partir de ce modèle
- Une classe possède un ensemble de variables, appelées **propriétés** ainsi qu'un ensemble de fonctions appelées **méthodes**.
- Il est possible de créer une classe à partir d'une classe grâce au mot-clé `new`. On appelle cela une **instance**.
- Pour accéder aux propriétés ou aux méthodes d'une classe, on utilise le symbole `->`.

### Créer des classes :

- Les objets possèdent des **propriétés** et des **méthodes**
- Syntaxe de base `< ?php class MaClasse {} ?>`
- Pour désigner une instance, nous utilisons le mot clé **\$this**, permettant de faire appel aux propriétés ou méthodes de notre objet au sein d'une autre méthode.

### Une API publique de vos objets

- Vous réduisez la complexité d'utilisation des classes en privatisant les méthodes et les propriétés avec `private`.
- Nous pouvons appliquer **le principe d'encapsulation**, en utilisant des getters et des setters, permettant d'accéder aux propriétés et de les modifier en s'assurant de garder des valeurs cohérentes.
- En réduisant la complexité, vous réduisez le risque d'erreur.

### Utilisez les propriétés et les méthodes statiques :

- En utilisant la staticité vous pouvez partager des valeurs, ou appeler des méthodes "utilitaires", n'ayant pas d'impact sur l'état de l'objet.
- Il est possible et recommandé de créer des constantes pour les informations qui ne changent pas, grâce au mot-clé `DEFINE`.
- Pour appeler une propriété ou méthode statique au sein d'une classe, il faut utiliser le mot clé **self**.

### Exploitez les méthodes communes aux objets :

- Il existe des méthodes dites magiques, qui sont appelées automatiquement par PHP. Elles commencent par `__`.
- `__construct` est la méthode appelée lors de la création d'une classe(un objet) . `__destruct` lors de sa suppression pour économiser de la mémoire.
- Il existe de nombreuses autres méthodes magiques disponibles [sur la documentation PHP](#).

### Les classes et l'héritage :


- L'héritage nous permet de passer les propriétés et méthodes d'une classe "parent" à des classes "enfants".
- Vous pouvez structurer votre code et éviter la duplication en héritant d'une classe.

- L'héritage s'effectue avec l'usage du mot clé `extends` juste après le nom de la classe à étendre, suivi de la classe dont il faut hériter.

### Profiter de l'héritage :

- La façon d'accéder aux propriétés des classes parentes s'effectue avec la flèche `->`.
- La façon d'accéder aux méthodes des classes parentes s'effectue également à l'aide de la flèche `->`.
- Lorsqu'il s'agit d'une méthode ou d'une propriété statique parente, l'accès s'effectue avec le mot clé **parent**.
- Vous pouvez réécrire une méthode existante dans un enfant, à condition de respecter sa signature.

### Contrôler l'accès aux propriétés et aux méthodes des objets :

- En utilisant le mot clé `private`, vous empêchez quiconque autre que la classe courante de manipuler les propriétés et méthodes.
- En utilisant le mot clé `protected`, vous donnez l'autorisation à la classe courante ainsi qu'à ses enfants de manipuler les propriétés et méthodes  ; mais pas aux éléments extérieurs à la classe, ou extérieurs aux objets de cette classe.
- Et comme nous l'avons vu précédemment, avec le mot-clé `public`, il est possible de manipuler la méthode ou la propriété depuis la classe, ses classes enfants et depuis l'extérieur.
- Autrement dit, pour modifier une propriété depuis un enfant, il faut qu'elle possède la visibilité `public` ou `protected`, ou qu'elle possède un mutateur.

### Contraindre l'usage des classes :

- En utilisant le mot clé **abstract**, vous pouvez imposer à une classe d'être héritée.
- Une classe abstraite ne peut plus être instanciée seule, et peut contenir des méthodes abstraites.
- Une méthode abstraite doit être implémentée dans les classes enfants, ou alors celle-ci doit aussi être abstraite.
- Vous pouvez également interdire l'héritage à l'aide du mot clé final **sur** une classe ou sur une méthode.

### Gérer le comportement d'une classe parente :

- Pour manipuler une constante et conserver la valeur de la classe courante, il faut y faire référence avec le mot clé **self**.
- Pour manipuler une constante et utiliser la valeur de la classe courante ou héritée si redéfinition, il faut y faire référence avec le mot clé **static**.

### Les namespaces :

- Vous pouvez avoir plusieurs classes du même nom en les cloisonnant dans un espace de noms.
- Déclarer un espace de noms s'effectue à l'aide du mot clé **namespace**.
- Un espace de noms peut être de plusieurs niveaux séparés par un `\`.
- Par défaut, PHP utilise un espace de noms global représenté par un `\`.

- Lorsque vous exploitez un espace de noms spécifique pour une classe, vous pouvez le déclarer pour tout l'espace de noms courant avec le mot clé **use**.

### La structure des projets :

- Les espaces de noms peuvent être utilisés pour correspondre à l'arborescence des fichiers, dans le but d'importer ces fichiers dynamiquement (PSR-4).
- Utiliser SPL combiné aux espaces de noms permet de charger les classes à la volée.
- Séparer les classes par fichiers réduit le nombre de fichiers à charger et à interpréter pour le langage.

### Les traits (héritage multiple) :

- Les traits vous offrent la possibilité d'étendre "horizontalement" votre code.
- Vous pouvez utiliser plusieurs traits dans une classe.
- Un trait peut être composé d'autres traits.

### Un contrat imposé avec les interfaces :

- Les interfaces vous offrent plus de souplesse et d'anticipation en permettant de :
  - o Typer des arguments et retours de méthodes qui n'existent pas encore ;
  - o Ou encore garantir qu'un code fonctionnera toujours sans s'imposer d'avoir à étendre des classes entières lors d'évolutions futures.

### Composition VS héritage :

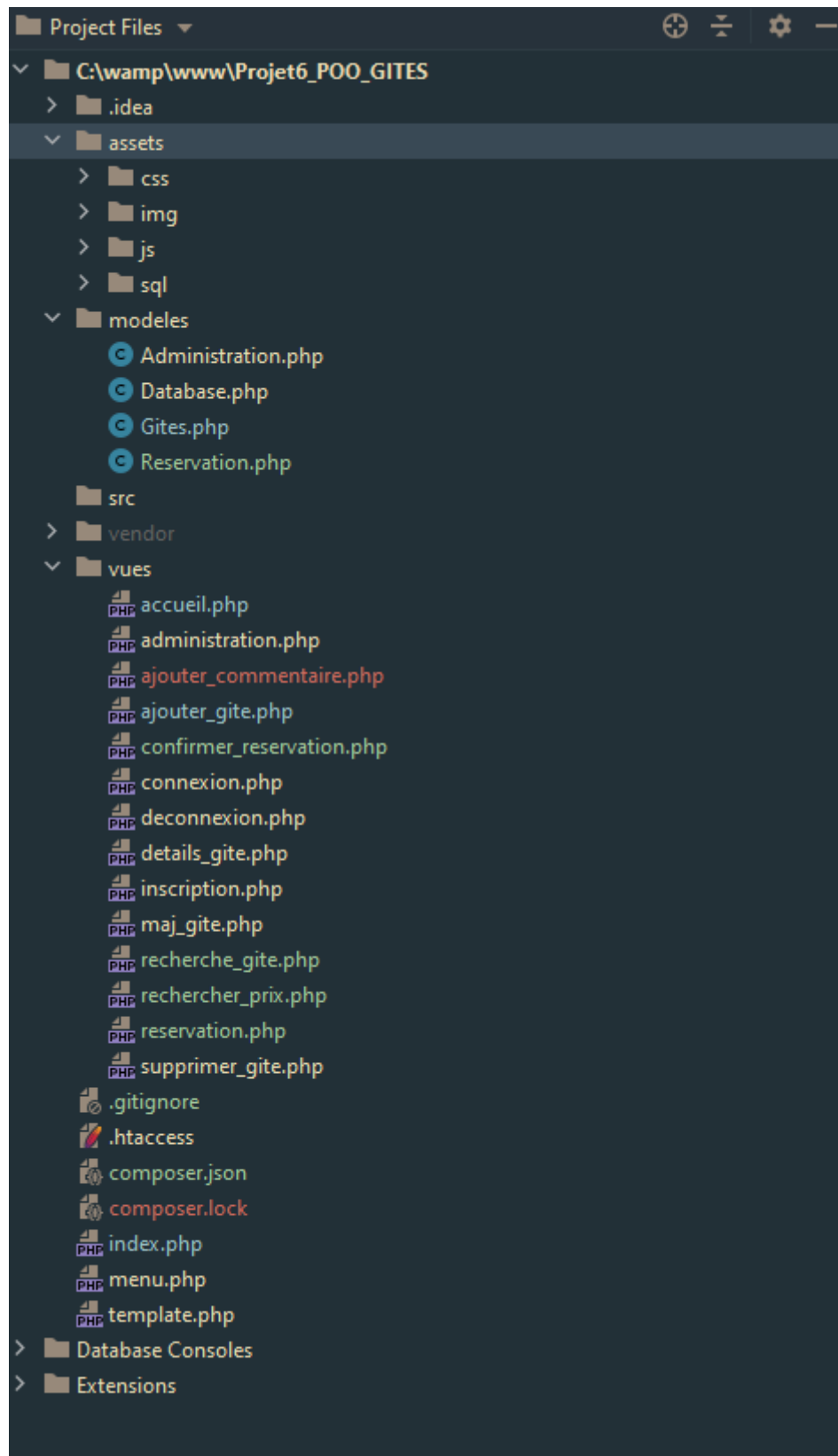
- Préférez la composition plutôt que l'héritage quand c'est possible.
- Pour savoir quand choisir l'un ou l'autre, posez-vous la question "est un"/"possède un".
- N'utilisez jamais les dépendances d'une dépendance directement, au risque d'avoir à reconstruire trop de choses après un simple changement.

### Gérer les erreurs avec try catch et exception :

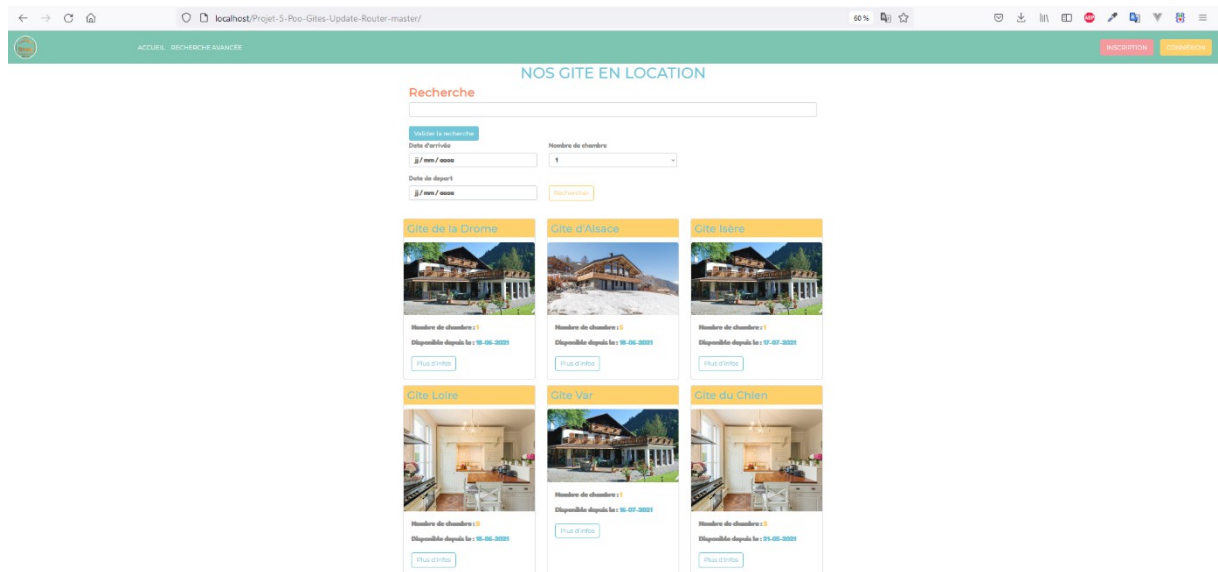
- PHP permet de gérer les erreurs avec les classes Exception en faisant remonter l'erreur à travers toute la pile d'exécution, grâce au mot clé throw.
- Les Exceptions offrent plus de finesse pour la gestion des erreurs avec le bloc try...catch.
- Créer vos exceptions vous donne plus de clarté dans votre code et dans le traitement des erreurs.
- Il est possible d'enchaîner les catches pour gérer les différents types d'erreurs.
- Le bloc finally placé après le(s) bloc(s) catch permet d'exécuter du code, qu'il y ait eu des erreurs, ou non.

### La structure du projet POO :





## La page d'accueil :



Cette étape de configuration du serveur apache permet la réécriture d'url :

<https://httpd.apache.org/docs/current/fr/howto/htaccess.html>

### 1. Configuration apache via le fichier .htaccess

```
index.php Database.php .htaccess
RewriteEngine on

RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.php?url=$1 [QSA,L]
```

Cette

étape de configuration du serveur apache permet la réécriture d'url :

Le router est un point d'entrée de votre application, il redirige toutes les URLs vers un fichier

L'extension `index.php?url='quelque chose'` utilise la variable super globale php `$_GET` → URL sera `projet/quelque chose`

L'extension `.php` disparaît et améliore le SEO (les crawlers index mieux les pages sans extension)

## 2. Le router et des URL propres

```
<?php
session_start();
ob_start();
//Utilisation de la variable superglobale $_GET['']

if(isset($_GET['url'])){
    $url = $_GET['url'];
}else{
    $url = "accueil";
}

if(!isset($_GET['url']) || empty($_GET['url'])){
}

if($_GET['url'] === ''){
    $url = 'accueil';
}

//Appel des vues
// url nom du dossier/index.php?url=accueil

if($url === 'accueil'){
    require 'views/accueil.php';
}elseif ($url === 'connexion'){
    require 'views/connexion.php';
}elseif ($url === 'deconnexion'){
    require 'views/deconnexion.php';
}elseif (isset($_SESSION['connecter']) && $_SESSION['connecter'] === true && $url === "administration"){
    require 'views/administration.php';
}elseif (isset($_SESSION['connecter']) && $_SESSION['connecter'] === true && $url === "ajouter_gites"){
    require "views/ajouter_gites.php";
}elseif ($url === "details_gite" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/details_gite.php";
}elseif (isset($_SESSION['connecter']) && $_SESSION['connecter'] === true && $url === "supprimer_gite" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/supprimer_gite.php";
}elseif (isset($_SESSION['connecter']) && $_SESSION['connecter'] === true && $url === "maj_gite" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/maj_gite.php";
}elseif ($url === "rechercher_gite"){
    require "views/recherche_gite.php";
}elseif ($url === "reservation" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/reservation.php";
}elseif ($url === "confirmer_reservation"){
    require "views/confirmer_reservation.php";
}elseif ($url === "inscription"){
    require "views/inscription.php";
}elseif (isset($_SESSION['connecter_user']) && $_SESSION['connecter_user'] === true && $url === "ajouter_commentaire" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/ajouter_commentaire.php";
}elseif ($url === "rechercher"){
    require "views/recherche_all.php";
}

elseif($url != '#:[\w]+#'){
    require 'views/404.php';
}

$content = ob_get_clean();
require "template.php";
```

PHP récupéré un paramètre placé dans URL,

En fonction de ce paramètre on appelle le fichier adapté :

exemple : ici on récupère le paramètre placé dans URL

```
if(isset($_GET['url'])){
    $url = $_GET['url'];
}else{
    $url = "accueil";
}
```

Si le paramètre `index.php?url='accueil'` → on appelle le fichier `accueil.php`

```
// url nom du dossier/index.php?url=accueil

if($url === 'accueil'){
    require 'views/accueil.php';
}
```

3. La mise en place d'un template générique
4. Utilisation de la fonction php ob\_start()
  - a. Cette fonction enclenche la temporisation de sortie
  - b. <https://www.php.net/manual/fr/function.ob-start.php>
  - c. ob\_start() démarre la temporisation de sortie. Tant qu'elle est enclenchée, aucune donnée, hormis les en-têtes, n'est envoyée au navigateur, mais temporairement mise en tampon.
  - d. Le contenu de ce tampon peut être copié dans une chaîne avec la fonction ob\_get\_contents(). Pour afficher le contenu de ce tampon, utilisez ob\_end\_flush(). Au contraire, ob\_get\_clean() effacera le contenu de ce tampon.
  - e. On place a la fin de notre router une variable \$content = ob\_get\_clean()

```
$content = ob_get_clean();
require "template.php";
```

- f. On appel le template php qui sera utilisé dans chaque page
- g. En gros le fichier template.php est utilisé comme gabarits de base de chaque page

**ici \$title et \$content sont des données dynamiques :**

```

<!doctype html>
<html lang="fr">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!--
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
  -->
  <link href="assets/css/bootstrap.css" rel="stylesheet">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?family=Fredoka+One&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="assets/css/styles.css">
  <title><?= $title ?></title>
</head>
<body>
  <header>
    <?php
      require "views/menu.php";
    ?>
  </header>
  <div class="container">
    <?= $content ?>
  </div>
  <script src="https://code.jquery.com/jquery-3.2.1.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
  <script src="assets/js/app.js"></script>
</body>
</html>

```

#### 4-BIS. La classe de connexion et sa méthode.

- La classe parente Database.php va être instanciée dans chaque fichier
- Chaque fichier aura donc accès à la méthode public getPDO()
- De cette manière on évite de réécrire plusieurs fois la connexion à PDO
- Noter que les variables (attributs ou propriétés) dans la classe, sont encapsulées et on a une visibilité interne à la classe
- Rappel : une propriété (variable) public est accessible de partout, private n'est accessible qu'au sein de la classe, protected est accessible dans la classe et celle dont elle hérite

## LA CLASSE ADMINISTRATION

```

class Database
{
    private $host = "localhost";
    private $dbname = "phpmvc";
    private $user = "root";
    private $pass = "";

    public function getPDO(){
        try {
            $db = new PDO( dsn: "mysql:host=".$this->host.";dbname=".$this->dbname.";charset=utf8", $this->user, $this->pass);
            $db->setAttribute( attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
            //echo "Connexion a PDO";
            //Retourne la propriété $db pour être utilisé dans autres fichier
            return $db;
        } catch (PDOException $e){
            echo "erreur de connexion " . $e->getMessage();
        }
        return null;
    }
}

```

5. Inscrire des utilisateurs (héritage de la classe mère Database.php pour la connexion a PDO) dans le dossier modeles/Administration.php
6. La classe Administration.php hérite de la classe Database.php pour avoir accès a la méthode getPDO()

```

<?php

//Appel du fichier de la classe de connexion
require "Database.php";

/**
 * Class Adminstration qui herite de database
 */
class Adminstration extends Database
{

```

7. 3 méthodes
  - a. Connecter un administrateur
  - b. Connecter les utilisateurs
  - c. Inscrire des utilisateurs (pour la réservation de gîte et ajouter des commentaires)

```

class Administration extends Database
{
    /**
     * @var string
     */
    private $email_admin;
    private $password_admin;

    /**
     * @var string
     *
     */
    //Pour utilisateur
    private $email_user;
    private $password_user;
    private $password_user_repeat;

    //Connexion des administrateur + accès au crud
    public function adminLogin(){...}

    //Connexion d'un utilisateur (non admin)

    public function userLogin(){...}

    //Ajouter un utilisateur qui poste des offres de location

    public function registerUser(){...}
}

```

## Formulaire d'inscription des utilisateurs

 [ACCUEIL](#) [RECHERCHE AVANCÉE](#) [INSCRIPTION](#) [CONNEXION](#)

### Inscription

Votre nom d'utilisateur

Votre nom d'utilisateur

Votre mot de passe

[S'inscrire](#)

[Retour](#)

Merci de remplir tous les champs du formulaire

Pour la connexion : 2 Rôles

 [ACCUEIL](#) [RECHERCHE AVANCÉE](#) [INSCRIPTION](#) [CONNEXION](#)

**Vous êtes :**

[Administrateur](#) [Client](#)

AU CLIC (jQuery ou Javascript) on affiche 2 formulaires sur 2 tables différentes

[ACCUEIL](#) [RECHERCHE AVANCÉE](#)

**Vous êtes :**

[Administrateur](#) [Client](#)

### CONNEXION A VOTRE ESPACE ADMINISTRATION

Email

Password

[Envoyer](#)

ICI UN SEUL ADMINISTRATEUR

### CONNEXION A VOTRE ESPACE CLIENT

Email

Password

[Envoyer](#)

ICI PLUSIEURS CLIENTS



- La vue appel : le fichier Administration.php
- On réalise une instance de la classe :
- Et on appelle la méthode adminLogin()

```

<?php
$title = "MIC GITES.COM -CONNEXION-";
require_once "modeles/Administration.php";
//Instance de la classe Admin
$admin = new Administration();

<?>
<h3 class="text-danger">Vous êtes : </h3>
<span>
  <a class="btn btn-outline-secondary" id="toggle-admin">Administrateur</a>
  <a class="btn btn-outline-info" id="toggle-user">Client</a>
</span>

<div id="form-admin">
  <?php

  if(isset($_SESSION['connecter']) && $_SESSION['connecter'] === true){
    header((string) "administration");
  }else{
    >
    <h2 class="mt-2 text-center text-warning">CONNEXION A VOTRE ESPACE ADMINISTRATION</h2>

    <form method="post">
      <div class="form-group">
        <label for="exampleInputEmail">Email</label>
        <input type="email" name="email_admin" class="form-control" id="exampleInputEmail" placeholder="Email">
      </div>
      <div class="form-group">
        <label for="exampleInputPassword1">Password</label>
        <input type="password" name="password_admin" class="form-control" id="exampleInputPassword1" placeholder="Mot de passe">
      </div>

      <button name="btn_valid_admin" type="submit" class="btn btn-primary">Connexion</button>
    </form>

    <?php
    if(isset($_POST['btn_valid_admin'])){
      $admin->adminLogin();
    }
  }
  >
</div>

```

- Si l'utilisateur est déjà connecté, on le redirige vers la page administration
- Sinon on affiche le formulaire
- On reproduit la même opération avec la table users

## LA CLASSE ADMINISTRATION

```

</div>
<div id="form-user">
  <?php
    if(isset($_SESSION['connecter_user']) && $_SESSION['connecter_user'] === true){
      header('Location: accueil');
    }else{
      ?>
      <h1 class="text-center text-secondary">CONNEXION A VOTRE ESPACE CLIENT</h1>

      <form method="post">
        <div class="form-group">
          <label for="exampleInputEmail1">Email</label>
          <input type="email" name="email_user" class="form-control" id="exampleInputEmail1" placeholder="Email">
        </div>
        <div class="form-group">
          <label for="exampleInputPassword1">Password</label>
          <input type="password" name="password_user" class="form-control" id="exampleInputPassword1" placeholder="Mot de passe">
        </div>

        <button name="btn_valid_user" type="submit" class="btn btn-secondary">CONNEXION</button>

      </form>

      <?php
        if(isset($_POST['btn_valid_user'])){
          $admin->usersLogin();
        }
      ?>
    }
  </div>

```

- Coté Administration (après connexion)
- Les méthodes de connexion :
- <https://github.com/michelonlineformapro/Projet-6-Poo-Gites-Update-Router/blob/master/Models/Administration.php>
- La classe Administration hérite de la classe mère Database et permet l'accès à la méthode public getPDO()
- Réalisé les opérations de CRUD :
  - d. Un modèle (classe = couche métier = SQL) modèles/Gites.php
  - e. Listes des méthodes à coder

```

1 <?php
2 //Appel du fichier class Database
3 require "Database.php";
4
5 //La classe Gites herite de la Classe Database => donc de ses attributs et methodes
6 class Gites extends Database{
7     //Creation des attributs privés de la classe Gites
8     private $id_gite;
9     private $nom_gite;
10    private $description_gite;
11    private $img_gite;
12    private $prix;
13    private $nbr_chambre;
14    private $nbr_sdb;
15    private $zone_geo;
16    private $disponible;
17    private $date_arrivee;
18    private $date_depart;
19    private $type_gite;
20
21    //Table commentaires
22    private $auteur_commentaire;
23    private $contenus_commentaire;
24    private $gites_id;
25    private $id_commentaire;
26
27
28    //Methodes public de liste des gites (client)
29    //Afficher tous les gites
30    public function getAllGiteAdmin(){
31    }
32    //Récupéré un gite par ID pour la page détails
33    public function getGiteById($id){
34    }
35    //Ajouter un gite appelé dans la page ajouter_gites.php
36    public function addGite(){
37    }
38    //Mise à jour des gites en mode administrateur
39    public function updateGite(){
40    }
41    //Supprimer un gite
42    public function giteToDelete($id){
43    }
44    //Confirmation de suppression du gite
45    public function deleteGite(){
46    }
47    //Afficher les gite disponible pour les visiteurs
48    public function availableGite(){
49    }
50    //Systeme de filtre par date a l'aide du formulaire de recherche
51    public function sortGiteByDate(){
52    }
53    //Méthode recapitulatif des données du gite réservé lorsque le visiteur valide la reservation dans l'email
54    public function recapGiteById($id){
55    }
56    //Le gite ne s'affiche plus lors de la confirmation de réservation
57    public function disabledGite(){
58    }
59    //Si la date du jour est supérieur à la date de départ et que disponible est = 0
60    public function checkDateGite(){
61    }
62    //Recherche de gite par mot clé dans nom_gite + description_gite + prix + category_gite
63    public function searchGiteByName(){
64    }
65    //Afficher des commentaires par gite
66    public function getCommentsByGite(){
67    }
68    //Ajouter un commentaire au gite
69    public function addCommentToGite(){
70    }
71 }

```

- Les vues (HTML + import du fichier modele + instance du modele + appel de la méthode concernée

## INSTANCE DE LA CLASSE GITES ET APPEL DE LA METHODE `getAllGiteAdmin()` :

1. Créer les attributs privés de la classe Administration

```

<?php
//Appel de la classe mere : Database.php connexion a PDO via le methode getPDO()
require_once "modeles/Database.php";
//La classe Gites herite de la Classe Database => donc de ses attributs et methodes
class Gites extends Database
{
    //Creation des attributs privés de la classe Gites
    private $id_gite;
    private $nom_gite;
    private $description_gite;
    private $img_gite;
    private $prix;
    private $nbr_chambre;
    private $nbr_sdb;
    private $zone_geo;
    private $disponible;
    private $date_arrivee;
    private $date_depart;
    private $type_gite;

    //Table commentaires
    private $auteur_commentaire;
    private $contenus_commentaire;
    private $gites_id;
    private $id_commentaire;
}

```

2. <https://github.com/michelonlineformapro/Projet-6-POO-Location-de-Gites/blob/master/modeles/Administration.php>

1. Afficher tous les gîtes (Admin)
2. Afficher les gîtes par ID
3. Ajouter un Gîte
4. Supprimer un Gîte
5. Mettre a jour un Gîte
6. Afficher le Gîte disponibles (tri par date)
7. Trier les Gîte par date de disponibilités
8. Réserver un Gîte
9. Vérifier les dates de disponibilité d'un Gîte
10. Rechercher un Gîte
11. Afficher les commentaires sur un Gîte
12. Ajouter les commentaires a un Gîte

## RESERVER UN GITE :

- Il s'agit d'une opération de base :
- L'utilisateur entre son email
- Un email est envoyé
- Dans l'email la requête SQL dans phpMailer passe l'attribut gites.disponible de true a false
- Lorsque que cet attribut est a false, il est afficher sur la page d'accueil, mais on ne peut plus accéder aux détails du gîte.
- Aucune date de réservation n'est demandée a l'utilisateur pour l'instant

- Créer un modèle (classe) Réservation (Modèles/Reservation.php)
- Ajouter et configurer PHP Mailer() et créer un compte mailTrap et – ou maildev
  - o <https://github.com/PHPMailer/PHPMailer>
  - o <https://mailtrap.io/>
- Le visiteur doit être connecter pour réserver un gîte

## INSTALLATION DE PhpMAILER

- Rendez-vous sur site packagist pour consulter les repositories PHP accessible via composer
- <https://packagist.org/>
- Installer composer : <https://getcomposer.org/download/>
- Dans votre projet : ouvrez un terminal et initialiser composer : `composer init`
- Repondre aux questions et valider
- Ceci a pour effet de générer un fichier composer.json dans lequel vous allez pouvoir installer des dépendances PHP
- Installer PhpMailer : `composer require phpmailer/phpmailer`
- Mettre a jour composer.json = `composer install`
- Créer une classe Reservation.php
- Ici on utilise l'héritage multiple a l'aide de PHP use

## Utilisation des espaces de noms : importation et alias

La capacité de faire référence à un nom absolu avec un alias ou en important un espace de noms est stratégique. C'est un avantage similaire aux liens symboliques dans un système de fichiers.

PHP peut aliaser(/importer) les constantes, fonctions, classes, interfaces, et les espaces de noms.

Un alias est créé avec l'opérateur use. Voici un exemple qui présente les cinq types d'importation :

### Exemple #1 importation et alias avec l'opérateur use

```
<?php
namespace foo;
use My\Full\Classname as Another;

// Ceci est la même chose que use My\Full\NSname as NSname
use My\Full\NSname;
```

- Auto charger des classes avec autoload.php du dossier vendor
- <https://coopernet.fr/formation/php/autoload>

```
configuration file contains list of composer dependencies
{
    "name": "micpi/projet6_poo_gites",
    "description": "POO PHP",
    "autoload": {
        "psr-4": {
            "Micpi\\Projet6PooGites\\": "modeles/"
        }
    },
    "authors": [
        {
            "name": "michel michael",
            "email": "m.michel@onlineformapro.com"
        }
    ],
    "require": {
        "phpmailer/phpmailer": "^6.6"
    }
}
```

- Ce fichier va charger automatique toutes les classes du dossier
- En haut de la classe Reservation.php

```
accueil.php x Gites.php x composer.json x Reservation.php x styles.scss x
<?php
//Import de la classe PHPMailer dans un namespace globale
//Ces appels doivent être en haut de page, pas dans une fonction
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;

//Appel de autoloader de classe
require "vendor/autoload.php";

class Reservation
{
    public function reserverGite(){
        //Instance de la classe phpmailer
        $mail = new PHPMailer();
        try {
```

- Pour réaliser nos test, nous avons besoin d'utiliser un catcher d'email, ici mailTrap :
- <https://mailtrap.io/>
- Une fois votre compte créer, récupérer les information de connexion pour PhpMailer

## My Inbox

SMTP Settings

Email Address

Auto Forward

Manual Forward

Team Members

SMTP / POP3 ?

Reset Credentials ↻

Use these settings to send messages directly from your email client or mail transfer agent.

⚠ Don't disclose your username or password as this may result in your inbox getting filled up with spam.

Show Credentials ▾

Integrations ?

cURL

- cURL
- Telnet
- Ruby**
  - Ruby on Rails
  - Ruby (net/smtp)
- Python**
  - smtpplib
  - Django
  - Flask-Mail
- PHP**
  - CakePHP 3.7+
  - CakePHP < 3.7
  - CodeIgniter
  - FuelPHP
  - Laravel 7+
  - PHPMailer**
  - Symfony 5+
  - WordPress
  - Yii Framework
  - Zend Framework
- Node.js

Congrats for sending test email with Mailtrap!

Inspect it using the tabs above and learn how this email can be improved.

```
p.io:2525' \
'aa6d7e0bd01' \
om \
i \
ample.com>
mple.com>
ternative; boundary="boundary-string"
charset="utf-8"
quoted-printable
e
```



- Ces informations sont uniques et vont se retrouver dans la configuration de votre méthode `reserverGite()`

```
public function reserverGite(){
    //Instance de la classe phpmailer
    $mail = new PHPMailer();
    try {
        //Config pour mailtrap
        // $mail->SMTPDebug = SMTP::DEBUG_SERVER; //Autorise le debug
        $mail->isSMTP(); //Utilisation du service mail transfer protocole
        $mail->Host = 'smtp.mailtrap.io'; //Appel du host mailtrap
        $mail->SMTPAuth = true; //Autorise et impose un user name + password
        $mail->Username = '1e9a0eeda636b9'; //Générer lors de la création du compte mailTrap = dans l'espace mailtrap roue crantée + smtp setting -> phpmailer
        $mail->Password = '64faad7e0bd01'; // Idem pour le mot de passe
        $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS; //La Transport Layer Security (TLS) ou « Sécurité de la couche de transport »
        $mail->Port = 2525; //Port pour mailtrap sinon -> 587 ou 465 pour 'PHPMailer::ENCRYPTION_SMTPS' et gmail
        $mail->setLanguage((langcode: 'fr', lang_path: '../vendor/phpmailer/phpmailer/language/'));
        $mail->CharSet = 'UTF-8';

        //Envoyeur et destinataire
        $mail->setFrom([address: 'locagite@gite.com', name: 'Annonces Administration']);
        $mail->addAddress([address: 'locagite@gite.com', name: 'Administrateur Annonces Games.com']);
        $mail->addReplyTo([address: 'locagite@gite.com', name: 'Annonces Administration']);
        //Connexion et requete PDO get by ID
        $user = "root";
        $pass = "";
        $db = new PDO([dsn: "mysql:host=localhost;dbname=phpmvc;charset=utf8;", $user, $pass]);
        $query = "SELECT * FROM gites INNER JOIN category_gites ON gites.gite_category = category_gites.id_category WHERE gites.id = ?";
        $req = $db->prepare($query);
        $id = $_GET['id'];
        $req->bindParam($parameter: 1, &variable: $id);
        $req->execute();

        //Contenu du mail
        $mail->isHTML( [isHTML: true]);
        $destinataire = $_POST['email_user'];
        $mail->Subject = "Validation de votre reservation du gite sur locagite@gite.com";
    }
}
```

- Dans une boucle while, on récupère l'id du gîte à réserver et on affiche ses détails dans un template HTML

```
while ($datas = $req->fetch()) {
    //Stock de l'id dans une variable
    $emailId = $datas['id'];
    //Url du liens de validation
    $url = "http://localhost/Projet-6-Poo-Gites-Update-Router/confirm_reservation?id=$emailId";
    //Contenus du mail
    $mail->Body = '

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta http-equiv="Content-Type" content="text/html">
    <title>Votre reservation chez locagite.com</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body style="background-color: #f0f0f0;">
<div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">

    
    <h3 style="text-align: center;">LOCA-GITES.COM</h3>
    <!--INFOS DE DEBUG -->
    <p>ICI URL DU GITE A RESERVER : ' . $url . ' </p>
    <p>ICI ID DU GITE A RESERVER : ' . $emailId . ' </p>
</div>
<div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">
    <h1>Loca-gite.com</h1>
    <h2>Vous : ' . $destinataire . '</h2>
    <p>Vous avez déposé une demande de reservation (ET C BIEN) avec le liens suivant</p>
    <p>Recapitulatif de votre commande</p>
    <p>Nom du gite :<b style="font-weight: bold;"> ' . $datas['nom_gite'] . ' </b></p>
    <p>Description du gite :<b style="font-weight: bold;"> ' . $datas['description_gite'] . ' </b></p>
    <p>Image du gite :</p>
    <p>Prix par semaine du gite :<b style="font-weight: bold;"> ' . $datas['prix'] . ' €</b></p>
    <p>Nombre de chambre :<b style="font-weight: bold;"> ' . $datas['nbr_chambre'] . ' </b></p>
    <p>Nombre de salle de bain :<b style="font-weight: bold;"> ' . $datas['nbr_sdb'] . ' </b></p>
    <p>Zone géographique :<b style="font-weight: bold;"> ' . $datas['zone_geo'] . ' </b></p>
    <p>Date arrivée :<b style="font-weight: bold;"> ' . $datas['date_arrivee'] . ' </b></p>
    <p>Date départ :<b style="font-weight: bold;"> ' . $datas['date_depart'] . ' </b></p>
    <p>Description du gite :<b style="font-weight: bold;"> ' . $datas['type'] . ' </b></p>
    <p>Toutes fois vous avez la possibilité d\'annuler ou de confirmer votre commande</p>
    <br />
    <a href=" ' . $url . ' " style="background-color: #007bff; color: white; padding: 5px 10px; text-decoration: none;">Confirmer la reservation de votre gite</a><br />
    <br />
    <p>Merci d\'utiliser notre site web</p>
    <p>Cordialement : Loca-gite.com: Michael MICHEL : Administrateur</p>
</div>
</body>
</html>
';

    $mail->body = "MIME-Version: 1.0" . "\r\n";
    $mail->body .= "Content-type:text/html;charset=utf8" . "\r\n";
}
```



- On stock dans une variable un lien de validation qui pointe vers notre site
- ```
$url = "http://localhost/Projet6_POO_GITES/confirmer_reservation&id=$emailId";
```
- Comme vous le constater ce lien est ajouter a un bouton dans le template HTML
- ```
<a href="' . $url . '" style="background-color: darkred; color: #F0F1F2; padding: 20px; text-decoration: none;">Confirmer la réservation de votre gîte</a><br />
```
- La requête pour désactiver le gîte se situe dans le fichier confirmer\_reservation&id=\$emailId
- Ce fichier appel de nouveau la classe Gites.php et la méthode disableGite()
- LES ETAPES :
- Pour réserver un gîte vous devez être connecter en tant qu'utilisateur

RECHERCHER

INSCRIPTION

Vous devez être un de nos clients pour réserver un gîte, merci de vous inscrire ou de vous connecter !

S'inscrire Connexion

- Un fois connexion on affiche la page reservation.php

ACCUEIL RECHERCHER Vous êtes connecté en tant que : **tito@test.fr**

INSCRIPTION DECONNEXION

Merci de remplir le formulaire avec votre email

## RÉSERVATION

Merci d'entrer votre email

Votre email@email.com

Reserver

- Enter votre email pour valider la réservation
- Une fois la requête effectuée, un message de succès apparaît

Un email viens de vous etre envoyé, merci de verifié votre boîte mail pour confirmer votre resevaton

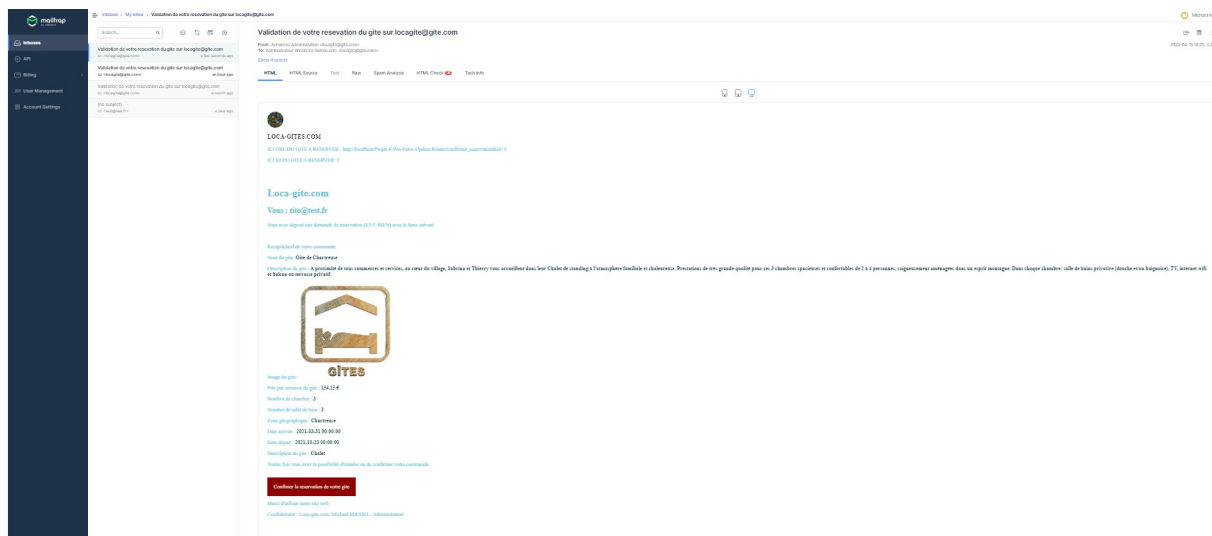
## RÉSERVATION

Merci d'entrer votre email

Votre email@email.com

Reserver

- Un fois la réservation effectuée mailTrap (via un liens) valide la réservation et redirige vers votre site



- Retour sur notre site et validation de la réservation

Votre réservation doit être validée !

### Récapitulatif de votre réservation :

[Retour à la page d'accueil](#)

Gîte de Chartreuse

Type : Chalet



#### Description :

A proximité de tous commerces et services, au cœur du village, Sabrina et Thierry vous accueillent dans leur Chalet de standing à l'atmosphère familiale et chaleureuse. Prestations de très grande qualité pour ces 3 chambres spacieuses et confortables de 2 à 4 personnes, soigneusement aménagées dans un esprit montagne. Dans chaque chambre: salle de bains privative (douche et/ou baignoire), TV, internet wifi et balcon ou terrasse privatif.

Nombre de chambre : 3

Nombre de salle de bains : 3

Zone géographique : Chartreuse

Prix à la semaine : 154,25 €

Votre demande est validée vous disposez d'un droit de retraction de 15 jours

[Condition générale de ventes \(CGV\)](#)

- Lors de la validation la méthode disableGite de la classe Gîtes modifie la date départ avec une requête SQL UPDATE
- *//Requête SQL de mise à jour le booléen disponible passe a 0 par gîte concerné*

```
$sql = "UPDATE gites SET date_depart = '2030-06-11 00:00:00' WHERE id_gite = ?";
```

- Sur la page d'accueil, le gîte réserver devient indisponible grâce a la méthode availableGite() de la classe Gîtes

## • LA RECHERCHE

- Créer une méthode searchGiteByName() dans votre classe Gîtes

---

### NOS GITE EN LOCATION

**Recherche**

Valider la recherche

Date d'arrivée

Nombre de chambre

1 ▾

Date de depart

Rechercher

- Pour la recherche texte : SQL

```
//Rechercher un gîte sur la page d'accueil avec des mot clés
//Recherche de gîte par mot clé dans nom_gîte + description_gîte + prix + category_gîte
public function searchGîteByName(){
    //Connexion à PDO
    $db = $this->getPDO();
    //Stocké la date du jour
    $today = date('format: "Y-m-d");

    //Recup de input recherche
    if(isset($_POST['recherche'])){
        $recherche = $_POST['recherche'];
    }else{
        $recherche = "";
        if(empty($recherche)){
            echo "<p class='alert-danger mt-2 p-2'>Merci d'enter un champ dans le barre de recherche</p>";
        }
    }

    $sql = "SELECT * FROM gites
            INNER JOIN categories ON gites.gîte_categorie = categories.id_categorie
            INNER JOIN regions ON gites.zone_geo = regions.id_region
            WHERE nom_gîte LIKE '%$recherche%' OR description_gîte LIKE '%$recherche%'
            OR prix_gîte LIKE '%$recherche%' OR gîte_categorie LIKE '%$recherche%' AND date_depart < '$today.'" AND disponible = 1";

    //Parcours des résultats
    $search = $db->query($sql);
    //var_dump($search);
}
```

### Exemple de requête SQL tri entre 2 date :

- Recherche par critères (Tranche de date + nombre de chambre)

```
//Systeme de filtre par date a l'aide du formulaire de recherche
public function sortGîteByDate(){
    $db = $this->getPDO();
    $today = date('format: "d-m-Y");

    //Récupération des valeurs du formulaire

    if(isset($_POST['date_arrivee'])){
        $date_start = $_POST['date_arrivee'];
    }
    if(isset($_POST['date_depart'])){
        $date_end = $_POST['date_depart'];
    }

    if(isset($_POST['nbr_chambre'])){
        $nbr_chambre = $_POST['nbr_chambre'];
    }
    /*
    var_dump($date_start);
    var_dump($date_end);
    var_dump($nbr_chambre);
    */
    //Ici on affiche les résultats ou les dates de départ dans la table sont inférieurs à la date entrée par l'utilisateur
    $search = $db->query(statement: "SELECT * FROM gites WHERE date_depart < '$date_end.'" AND nbr_chambre = '$nbr_chambre.'"");
    //var_dump($search);
}
```

- La méthode sortGîteByDate() de la classe Gîtes

### Rechercher un gîte par prix minimum par nuit

- La méthode sortGîteByPrices() de la classe Gîtes

```

public function searchGitesByPrices(){
    $db = $this->getPDO();

    $prix_null = 0;
    $prix_min = 10;
    $prix_med = 100;
    $prix_med2 = 500;
    $prix_high = 1000;

    if(isset($_POST['prix']) && !empty($_POST['prix'])){
        $this->prix = $_POST['prix'];

        if($this->prix == 1){
            $this->prix = $prix_null;
        }
        if($_POST['prix'] == 2){
            $this->prix = $prix_min;
        }
        if ($_POST['prix'] == 3){
            $this->prix = $prix_med;
        }
        if ($_POST['prix'] == 4){
            $this->prix = $prix_med2;
        }
        if ($_POST['prix'] == 5){
            $this->prix = $prix_high;
        }
    }

    var_dump($this->prix);

    $today = date(format: "Y-d-m");
    $sql = "SELECT * FROM gites INNER JOIN categories ON gites.gite_categorie = categories.id_categorie WHERE prix_gite >= ?";
    $req = $db->prepare($sql);

```

## AJOUTER DES COMMENTAIRES

- L'utilisateur doit être connecter pour ajouter un commentaire

## DÉTAILS DU GITE

Gite Haute-Alsace

Type : Igloo



RESERVER

RETOUR

Ajouter un commentaire

### Description :

L'incontournable Route des Vins d'Alsace qui déroule son itinéraire à travers le vignoble sur 170km du nord au sud de la région. Venez déguster nos vins blancs : Sylvaner, Pinot blanc, Riesling, Muscat, Pinot gris, Gewurztraminer, soit 51 Grands Crus pour les amateurs de viticulture. Visitez les villages traditionnels comme Saint-Hippolyte, village reconnu pour son vin rouge, Ribeauvillé avec sa fête des Ménétriers et autres petits villages comme Riquewihr ou Kaysersberg, reconnus pour leurs marchés de Noël. Vous pourrez prolonger votre expérience au cœur des rites, coutumes et légendes de la région au cours d'une journée immersive à l'Écomusée d'Alsace.

Nombre de chambre : **2**

Nombre de salle de bains : **3**

Zone géographique : **Guyane**

Prix à la semaine : **452.25 €**

OUI

Disponible : **OUI**

- Ajout d'un formulaire de commentaire

Bonjour : **tito@test.fr**

## Ajouter un commentaire

tito@test.fr

Votre commentaire :

Ajouter un commentaire

Merci remplir tous les champs !

- Depuis la vue appel de la méthode addCommentToGite() de la classe Gîtes

```

//Ajouter un commentaire au gite par utilisateur connecté
//Ajouter un commentaire au gite
public function addCommentToGite(){
    //Connexion a pdo
    $db = $this->getPDO();
    //Recup des elements du formulaire
    //On verifie les champs du formulaires
    if(isset($_POST['auteur_commentaire'])){
        $this->auteur_commentaire = $_POST['auteur_commentaire'];
    }else{
        echo "<p class='alert-danger p-2'>Merci de remplir le champ auteur du commentaire</p>";
    }

    if(isset($_POST['contenus_commentaire'])){
        $this->contenus_commentaire = $_POST['contenus_commentaire'];
    }else{
        echo "<p class='alert-danger p-2'>Merci de remplir le champ contenu du commentaire</p>";
    }

    if(isset($_POST['gites_id']) && !empty($_POST['gites_id'])){$
        $this->gites_id = $_POST['gites_id'];
    }else{
        echo "<p class='alert-danger p-2'>Merci de remplir le champs !</p>";
    }

    //Ici gites_id sera un champs caché et prendra l'id du gite passé dans URL
    $sql = "INSERT INTO commentaires (auteur_commentaire, contenus_commentaire, gite_id) VALUES (?,?,?)";
    $insert = $db->prepare($sql);
    $insert->bindParam( parameter: 1, &variable: $this->auteur_commentaire);
    $insert->bindParam( parameter: 2, &variable: $this->contenus_commentaire);

    $insert->bindParam( parameter: 3, &variable: $this->gites_id);
    $insert->execute(array(
        $this->auteur_commentaire,
        $this->contenus_commentaire,
        $this->gites_id
    ));
}

```

- La vue ajouter\_commentaire.php

```

<?php
require_once "modeles/Gites.php";
//Instance de la classe gite
$gites = new Gites();

if(isset($_SESSION['connecter_user']) && $_SESSION['connecter_user'] === true){

    ?>
    <h4 class="text-danger mt-1"><b>Bonjour :</b> <?= $_SESSION['email_user'] ?></h4>
    <?php
    $email = $_SESSION['email_user'];
    $id = $_GET['id'];
    ?>
    <div class="main-container mt-3">
        <h1 class="text-info text-center">Ajouter un commentaire</h1>
        <form method="post">

            <div class="form-group">
                <input type="text" value="<?php $email ?>" class="form-control" name="auteur_commentaire" placeholder="<?= $_SESSION['email_user'] ?>">
                <?php
                $_POST['auteur_commentaire'] = $email;
                //var_dump($email);
                ?>
            </div>

            <div class="form-group">
                <label for="contenus_commentaire">Votre commentaire : </label>
                <textarea class="form-control" id="contenus_commentaire" name="contenus_commentaire" rows="5"></textarea>
            </div>

            <div class="form-group">
                <input type="hidden" value="<?= $id ?>" name="gites_id">
            </div>

            <button type="submit" name="btn-add-comment" class="btn btn-outline-success">Ajouter un commentaire</button>
        </form>
    </div>

    <?php

    if(isset($_POST['btn-add-comment'])){
        $gites->addCommentToGite();
    }else{
        echo "<p class='alert alert-warning p-3 mt-3'>Merci remplir tous les champs !</p>";
    }

}

}

echo "<p class='alert alert-danger p-3 mt-3'>Merci de vous connectez pour posy   un commentaire</p>";
}
}

```

- Pour afficher les commentaires de chaque utilisateur on utilise la méthode `getCommentsByGite()` de la classe `Gîtes`

```
//Afficher des commentaires par gite
public function getCommentsByGite(){
    //Connexion à PDO
    $db = $this->getPDO();
    //Requête SQL selection de tous depuis la table commentaires ou la clé étrangère est a ID du gite trié par l'ID du commentaire
    $sql = "SELECT * FROM commentaires INNER JOIN gites ON commentaires.gite_id = gites.commentaire_id WHERE commentaires.gite_id = ?;";
    //Requête préparée
    $req = $db->prepare($sql);
    //Liaison récupération de l'ID dans URL
    $req->bindParam(1, $this->GET['id']);
    //Execution de la requête
    $req->execute();
    //Départ de la liste
    ?>
    <ul class="list-group mt-2">
        <li class="list-group-item active">Commentaire : </li>
        <?php
        //Boucle de lecture des commentaires
        foreach ($req as $row){
            //Si on a des commentaires par ID
            if($row){
                ?>
                <li class="list-group-item">Nom de l'auteur : <b class="text-info"><?= $row['auteur_commentaire'] ?></b></li>
                <li class="list-group-item">Commentaire de l'auteur : <b class="text-info"><?= $row['contenus_commentaire'] ?></b></li>
                <br>
                <?php
            }else{
                echo "<p class='alert-danger p-2 mt-2'>Aucun commentaire pour de gite</p>";
            }
        }
        ?>
    </ul>
    <?php
}
```

- Dans la page `details_gite.php` : on affiche les commentaires par id du gîte concerné

Commentaire :
Nom de l'auteur : <b>Michael25</b>
Commentaire de l'auteur : <b>Ce gite est très sympa dans un cade bucolique et le proprio est cool</b>
Nom de l'auteur : <b>Michael25</b>
Commentaire de l'auteur : <b>Ce gite est très sympa dans un cade bucolique et le proprio est cool</b>
Nom de l'auteur : <b>Michael25</b>
Commentaire de l'auteur : <b>Ce gite est très sympa dans un cade bucolique et le proprio est cool</b>
Nom de l'auteur : <b>Michael25</b>
Commentaire de l'auteur : <b>Ce gite est très sympa dans un cade bucolique et le proprio est cool</b>
Nom de l'auteur : <b>Test</b>
Commentaire de l'auteur : <b>vous permettant de cuisiner en libre accès. Salon commun accessible à tous les hôtes avec cheminée à l'ancienne, Tv, bibliothèque, Internet wifi.</b>

**BRAVO VOUS CONNAISSEZ DESORMAIS LA POO PHP**



